AFRL-IF-RS-TR-2004-297
In-House Report
October 2004

A VISUAL DATA HASH METHOD

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

STINFO FINAL REPORT

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2004-297 has been reviewed and is approved for publication

APPROVED: /s/

GERALD C. NETHERCOTT, Chief
Multi-Sensor Exploitation Branch
Information & Intelligence Exploitation Division
Information Directorate

FOR THE DIRECTOR: /s/

JOSEPH CAMERA, Chief
Information & Intelligence Exploitation Division
Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	OCTOBER 2004	In-House Apr 02 – Sep 04

4. TITLE AND SUBTITLE
A VISUAL DATA HASH METHOD

5. FUNDING NUMBERS
C - N/A
PE - 62702F
PR - 459E
TA - H0
WU - C1

6. AUTHOR(S)
Andrew J. Noga

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory/IFEC
525 Brooks Road
Rome New York 13441-4505

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory/IFEC
525 Brooks Road
Rome New York 13441-4505

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
AFRL-IF-RS-TR-2004-297

11. SUPPLEMENTARY NOTES

AFRL Project Engineer: Andrew J. Noga/IFEC/(315) 330-2270/ Andrew.Noga@rl.af.mil

12a. DISTRIBUTION / AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 Words)*

This report describes the Detector and Extractor of Fileprints (DEF) process for data protection and automatic file identification. The DEF process generates binary images from an arbitrary data file that are unique to the data file. These visual hashes, or fileprints as referred to herein, are based on spectral characteristics of the data. The DEF employs the patented Adjustable Bandwidth Concept Signal Energy Detector.

14. SUBJECT TERMS
Data Fingerprint, Fileprint, Visual Hash, Adjustable Bandwidth Concept

15. NUMBER OF PAGES
23

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Table of Contents

List of Figures

1.0 Introduction

In this in-house research effort, several signal processing issues have been addressed as they apply to the mission of the Information Exploitation Division. Focus has been on the application of the Adjustable Bandwidth Concept (ABC) signal pre-detection processor [1,2]. In particular, this report will present a novel method of data protection which employs the ABC process. For additional information on other related research under this effort, the reader is referred to [3,4].

In this report an invention is described for data protection, which is currently in the status of patent pending. The purpose of the invention is to provide a customizable, efficient and effective method of measuring the integrity of a data sequence of interest. Integrity is lost when bit changes occur within a data sequence to be protected. Changes are typically due to transmission errors or due to tampering. It is also a purpose to include a provision for assessing the extent and general location of the changes within the data. Of particular interest are data sequences that are of substantial size, for example, on the order of 10,000 bytes or more. Although it is sometimes possible to make comparisons to a protected duplicate for integrity measurement, it becomes impractical and difficult to make comparisons when the data sequence size and the number of data sequences are substantial.

An additional purpose of the invention is to provide a method of measuring similarity between data sequences. Similarity measurement can be used for demonstrating ownership of data. In this context, in addition to an ability to measure integrity, what is desired is an ability to conclude whether or not the data of interest has been derived through alteration of previously existing data. This can also allow for data authentication, which is the determination of the origins of the data.

Yet another purpose of the invention is to provide a method of automatically identifying the data type. For example, when packets of data are available prior to knowledge of the source, determining the data type can be useful for sorting. In the case of data files, this can be an ability to determine the filename extension (such as .bmp or .wav), without prior knowledge of the filename.

2.0 Background

Methods exist for processing an original data sequence in order to generate information about the data for the purposes of integrity measurement, ownership demonstration and authentication. Background will be given on three categories of methods, each of which addresses subsets of these purposes. The first category is digital watermarking, the second is data hashing and the third is error detection coding. Although the subject invention does not involve the embedding of watermarks, watermarking is often used for the ownership demonstration and authentication purposes of the subject invention. The invention is better categorized as a data hash. Data hashers are often used for integrity measurement purposes. Background regarding hashing is given, after an introduction to digital watermarking. This section concludes with background on error detection coding. As with data hashing, the purpose of error detection coding is to measure data integrity. However, it is normally done as part of data transmission protocol and therefore addresses transmission errors rather than tampering.

An important step in data authentication and demonstrating data ownership is some form of registration process. This is a process that is recognized and accepted by the community of users whereby information regarding the original data is stored and later presented for comparisons. It is often important that this information is also time-stamped, properly associated with the original data and stored in a secure fashion. This identifying information regarding the original data can be referred to as registration data. In the exemplifying case of digital watermarking, the watermark, the watermark embedding method and the watermark recovery method can become part of the registration data.

2.1 Digital Watermarking

For a more in-depth review of digital watermarking, the reader is referred to [5] and [6]. Digital watermarking is the process of embedding identification data within host data typically for authentication of the host data, and for demonstrating ownership of the host data. Other applications exist and are given in the cited reference. Often, the goal is to embed the identification data in such a manner that changes are imperceptible to a human observer, when the observer only has access to the resulting watermarked data. The identification data or watermark is presumably a small amount of data relative to the host data, making the task of hiding the watermark less difficult. For example, small changes to a group of adjacent pixels in an image can be made in an imperceptible fashion when knowledge of the human ability to detect differences in color and or intensity are taken into account. Likewise, audio data can host imperceptible changes by applying an understanding of the human auditory response. Thus to a human, the watermarked audio data can sound exactly like the original, but in fact be slightly different.

When the goal is to embed watermarks in an imperceptible fashion, there is a trade-off that exists between imperceptibility and robustness to attack. Basically, as more changes are made to the host data as a result of embedding the watermark, better protection can be obtained. Consequently, it becomes less likely that the resulting watermarked data can be altered by an attacker in a manner such that the watermark is essentially removed and the post-attack watermarked data is still useful. However, as more data is embedded, a point is reached where the watermarked data is perceptibly different from the host data. A balance must be obtained such that a margin of robustness to attack and watermarked data quality is maintained.

There are two major disadvantages to digital watermarking. The first is with regard to the fact that a process is required for embedding the watermark within the host data. This process will take time and will require computational resources. The second disadvantage is with regard to the fact that embedding a digital watermark within the host data will result in an alteration of the host data. This is undesirable when the host data is, for example, a digital image or a digital audio recording. In such cases, the watermarking can adversely affect image and audio quality. Even when a human cannot perceive the changes caused by watermarking, the resulting changes may reduce the value of the data.

2.2 Data Hash

A data hash is typically used for determining if one or more bits within a data sequence have changed. More recently, hashing has been proposed as an alternative to digital watermarking, when the interest is to provide a method of ownership demonstration and authentication [7]. As the term implies, a data hash is a repeatable process by which the original data is reorganized and reduced to a short sequence for protection purposes. This short output data sequence has been referred to as a fingerprint, a message digest, a hash value or simply a hash of the input data sequence. As with human fingerprints and humans, a hash value is claimed to be with high probability unique to a given input data sequence. In contrast to digital watermarking, it is not necessary for a data hash to embed auxiliary data within the original host data sequence.

Given only the hash value and hash process, it would require a search and hash procedure over an available set of data sequences to determine if the hash value came from a particular sequence. This would make it impractical for an attacker to obtain the original sequence based only on knowledge of the hash value and hash process. Alternatively, a hash value, hash process and input data sequence can be made public. This can help assure the recipients of distributed data sequences that the data is authentic and integrity has been maintained. As an example, after downloading a file over a network, the recipient can first run the hash process on the file to generate the hash value. If this hash value matches the known hash value, then the file can be considered free of unauthorized alterations.

A generic description of a data hash process is shown in Figure 1. Pre-processor 101 and Post-processor 103 are optional steps. Processes for Pre-processor 101 can include transformation into an alternate representation domain, data padding and statistical calculations. These operate on the Input Data Sequence which is to be protected. For example, a data hash process may transform data sequences from the time domain to the frequency domain. Some hashes require the appending of known data such as a sequence of zeros in order to establish sequences of specific size. Post-processor 103 can include encoding and compression. These steps operate on the output of Hasher 102 and are often necessary in practical applications. The Output Data Sequence from Post-processor 103 is a representation of the hash of the Input Data Sequence. In most hash processes, the hash value is represented in the Output Data Sequence as a fixed length alphanumeric using hexadecimal notation.

Hasher 102 operates on the output sequence from Pre-processor 101, in such a manner as to reduce the data to an essentially unique hash value. For example, a hash process may generate a 128 bit hash value for each Input Data Sequence. This will allow for 2^{128} hash values and potentially as many unique Input Data Sequences.

Secure Key 104 contains the parameters necessary for the operation of components 101, 102, and 103. Pre-process parameters may include values for data segmentation, indexing and transformation to other representation domains. Post-processing parameters may include control information for encoding or compression processes. Subsets of the various parameters can include seed values for the generation of pseudo-random number sequences. The use of pseudo-random sequences contributes to the security of the overall hash process.

An advantage that data hashing has over digital watermarking is that no embedding process is required. This preserves the quality of the original input data sequence and is particularly important for digital images and digital audio. However, the typical data hash process has the disadvantage that even though unauthorized alterations of one or more bits will likely be detected, the resulting hash value yields no information about the nature of the alterations.

Information is not obtained such as how many bit alterations have taken place, where such changes have occurred within the data sequence and how similar the altered data is to the original data. This is information that would be necessary for ownership demonstration and authentication. Newly proposed hash methods have begun to address these disadvantages for the case of image data using a wavelet decomposition [7]. Hash methods that address these disadvantages and are more generally applicable to a variety of data sequences are required.

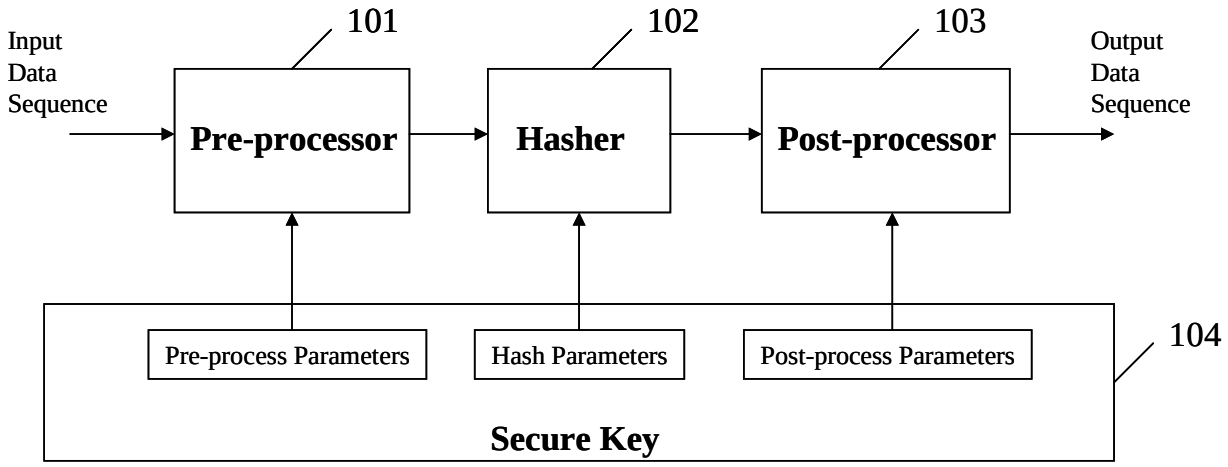

Figure 1. Prior methods of data hashing.

2.3 Error Detection Coding

Error detection coding is a mature technology area within the field of digital communications and data transmission. As with data hashing, the purpose of error detection coding is to measure data integrity. Unlike data hashing, the coding can be designed to allow for the further purpose of error correction. Error detection coding is normally done as part of data transmission protocol and addresses transmission errors rather than tampering. These transmission errors occur as a result of additive noise and other distortions present in the transmission channel. Reference is made to [8] for details regarding coding for the purpose of detecting transmission errors in received data sequences. Some background is provided here for contextual purposes in relation to the subject invention.

In error detection coding, data sequences to be transmitted are mapped to new data sequences prior to transmission. The mapping results in a substantial increase in the size of the transmitted data sequence. Error detection capability is achieved at the expense of this increase in size. For example, some error coding techniques operate on blocks or segments of consecutive bits that comprise the entire data sequence. In this case, each segment of sequential bits of information is mapped to what is referred to as a code word. The mapping is done in such a way that each code word contains more bits than the corresponding information segment. Each code word is also a sequence of bits that uniquely corresponds to one of the possible information segments. However, because the code words consist of more bits, a designer can choose code words that are more easily separated at the receiver. Once received, the code words are simply mapped

back to the original information segment. Without the error coding, the shorter length information segments are more difficult to distinguish between at the receiver, potentially causing transmission errors.

Error detection coding has some advantages over most data hash and watermarking processes. Although not all errors are detected, the coding often helps to identify segments of the data sequence that contain errors. Rather than re-transmitting the entire data sequence, this allows for re-transmission requests for only the affected segment. Also, with error detection coding, errors can often be corrected with no need for re-transmission. A disadvantage of error detection coding is the increase in size of the transmitted data sequence. This leads to an increase in the transmission rate, and therefore requires additional bandwidth or power resources. It is not uncommon for error detection coding schemes to result in a doubling of the data sequence size. A further disadvantage of error detection coding is that the purposes of ownership demonstration and authentication are not achieved. For example, unauthorized alterations made prior to transmission will result in a modified data sequence that is treated by the error coding process as any other input. Thus the coding process will attempt to faithfully transmit the modified data sequence.

3.0 Detector and Extractor of Fileprints

The Detector and Extractor of Fileprints (DEF) apparatus is a novel method of processing data sequences for the purposes of data integrity assessment, data ownership demonstration and data authentication. The DEF process operates on an input data sequence to extract what could be called naturally occurring digital watermarks. More appropriately, these naturally occurring digital watermarks are referred to herein as fileprints. As implied, fileprints are data sequences that have been extracted from the input data sequence in a repeatable manner, and are with high probability unique to a given input. In this sense, a fileprint can be used to identify which data sequence it came from, just as a human fingerprint can identify a particular person. Because the fileprints have not been embedded, they are not actually digital watermarks. However, fileprints can be used in a like fashion to digital watermarks for information protection.

A block diagram of the subject DEF invention is given in Figure 2. The DEF accepts an Input Data Sequence for processing by Segment Selector 201. Based on parsing parameters from Secure Key 206, the Input Data Sequence is partitioned by Segment Selector 201 into a set of segments with L-samples per segment. Here, a sample is a sequence of bits that can be interpreted as a number. Each segment of the set is processed in turn by Periodogram Calculator 202, which generates an estimate of the power spectral density of each segment. For a given input segment, the output of Periodogram Calculator 202 will be an N-sample segment, where N is no larger than L. Log-scaled estimates are particularly effective, as when the discrete-Fourier-transform of the data is used and the log-magnitude of the transform is retained. Each resulting segment is input to ABC Processor 203. Detail regarding the processing steps of ABC Processor 203 can be found in U.S. Patent 5,257,211. ABC parameters such as the number of stages, M, are obtained from Secure Key 206. M fileprints are generated by Threshold-Based Detector 204. This is accomplished by comparing the ABC Processor 203 outputs to thresholds from Secure Key 206. Thus if the parsing of the Input Data Sequence results in a set containing S segments, each fileprint associated with the input will be a binary array of size S rows by N columns. For

efficient storage, retrieval, transmission and utilization, the fileprints can be reduced in size by Compressor/Encoder 205 such that representations for R of the M fileprints are available. The parameters for Compressor/Encoder 205 can optionally be stored in Secure Key 206, or stored separately as shown in the figure. The output of Compressor/Encoder 205 is a set of R compressed and/or encoded fileprints, where R is no larger than M.

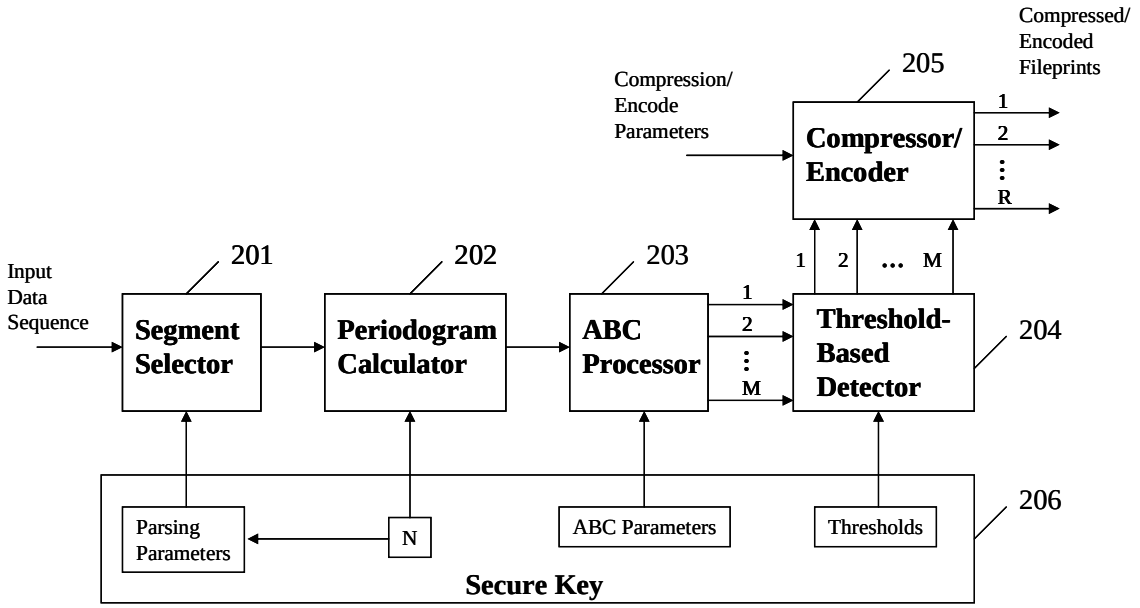

Figure 2. The Detector and Extractor of Fileprints.

Due to the user-selectable parameters of Secure Key 206 along with the compression/encode parameters, the DEF apparatus allows for a highly customizable method of generating fileprints. In addition, the parameters of Secure Key 206 are retained in a method appropriate for a desired level of secrecy regarding how the fileprints have been generated. The typical parameter selection process will begin with the choice of the integer, N. Selecting N to be a fixed value and a power of 2 will allow for simplified processing in Periodogram Calculator 202 using the popular fast-Fourier transform (FFT). Once N is selected, parsing parameters can be chosen. The parsing parameters for Segment Selector 201 determine how the S subsets of data are formed from the Input Data Sequence. This includes how to interpret the Input Data Sequence as a sequence of samples. For example, the Input Data Sequence can be interpreted as a sequence of 8-bit unsigned integer-valued samples. This can help increase sensitivity to unauthorized changes, relative to the choice of greater than 8 bits per sample. Note that in the case where the Input Data Sequence represents samples such as audio amplitudes or image pixel colors, Segment Selector 201 is not constrained to the same interpretation of samples. Another function of Segment Selector 201 is to produce the L-sample segments from specified starting locations within the Input Data Sequence. This allows for the selection of segments with overlapping sample boundaries. These starting locations are among the parsing parameters.

The remaining parameters to be chosen by the user are the ABC and threshold parameters. The ABC process allows for an M-stage approach to averaging over frequency and over time. By considering the output of Periodogram Calculator 202 to be time-consecutive segments, the ABC parameters can be chosen accordingly. Although a large range of thresholds will give useful results from Threshold-Based Detector 204, fileprint characteristics are directly affected by the choice of thresholds. The preferred mode of operation is obtained when each of the binary fileprint arrays contains a 1 in approximately 50 percent of the elements of the arrays. The number of elements in a fileprint is the product of S and N. To achieve this characteristic for fileprint 2 through M, threshold values close to zero can be chosen. To achieve this characteristic for fileprint 1, statistics of the samples from stage 1 of ABC Processor 203 can be used to set the threshold for stage 1. Regardless of the manner in which the thresholds are chosen, these and all other parameters must be stored to allow for the ability to reliably reproduce the fileprints.

3.1 An Example DEF Application

An example application of the DEF apparatus is provided in Figures 3(a) and 3(b). Note that the example is not intended to be representative of the best application. The example is provided with the intent to aid in the understanding of the DEF process.

In the example, an Input Data File is to be stored on Storage System/Network 304 as shown in Figure 3(a). At the same time, fileprints extracted from the Input Data File using DEF 306 are stored on Storage System/Network 307. For this example, the original Input Data File is processed by Compressor/Encoder 301 to make available a compressed/encoded version of the file. A switch, SW 302, determines whether or not the Input Data File or the compressed/encoded version is sent to Router 303 for potential storage. In this example, lossless compression is assumed. Likewise a switch SW 305 determines whether fileprints will be extracted from the Input Data File or the compressed/encoded version. Under control of Storage-Retrieval-Registration-and-Verification-Approver 308, the status of SW 305, the fileprints, DEF parameters and Input Data File identification such as name and date are stored on Storage System/Network 307. Likewise, under control of Storage-Retrieval-Registration-and-Verification-Approver 308, the status of SW 302, the version of the file from Router 303 and Input Data File identification such as name and date are stored on Storage System/Network 304. The dashed lines around SW 305, DEF 306 and Storage System/Network 307 are used to imply a higher level of security for these devices and data paths. This completes the description of the data file storage process for the example DEF application.

Continuing with the example, reference is made to Figure 3(b) depicting file retrieval. To retrieve an Input Data File previously stored by the system of Figure 3(a), a request for retrieval is made by providing some identification aspect of the file such as file name. The file and status of SW 302 are retrieved from Storage System/Network 304 and made available to Decompressor/Decoder 310. At the same time, the status of SW 305, the fileprints, and DEF parameters associated with the file are retrieved from Storage System/Network 307. Parameter Extractor 312 directs the retrieved DEF parameters to DEF 309 and the retrieved status of SW 305 to Decompressor/Decoder 310. Based on the status of SW 305, Decompressor/Decoder 310 will output either the compressed/decoded version of the stored file, or the original stored file to

DEF 309. Likewise, based on the status of SW 302, Decompressor/Decoder 310 will output either the compressed/decoded version of the file, or the original file to Verifier 311. Verifier 311 also receives from DEF 309 the extracted fileprints from the stored file. With these newly generated fileprints which have been extracted, Verifier 311 compares these to the fileprints retrieved from Storage System/Network 307. Similarity measurements made by Verifier 311 during the compare process are sent to Storage-Retrieval-Registration-and-Verification-Approver 308, which then directs Verifier 311 to provide the Output Data File, if appropriate similarity measurements have been obtained. The dashed lines around Decompressor/Decoder 310, DEF 309, Verifier 311, Parameter Extractor 312 and Storage System/Network 307 are used to imply a higher level of security for these devices and data paths. This completes the description of the data file retrieval process for the example DEF application.

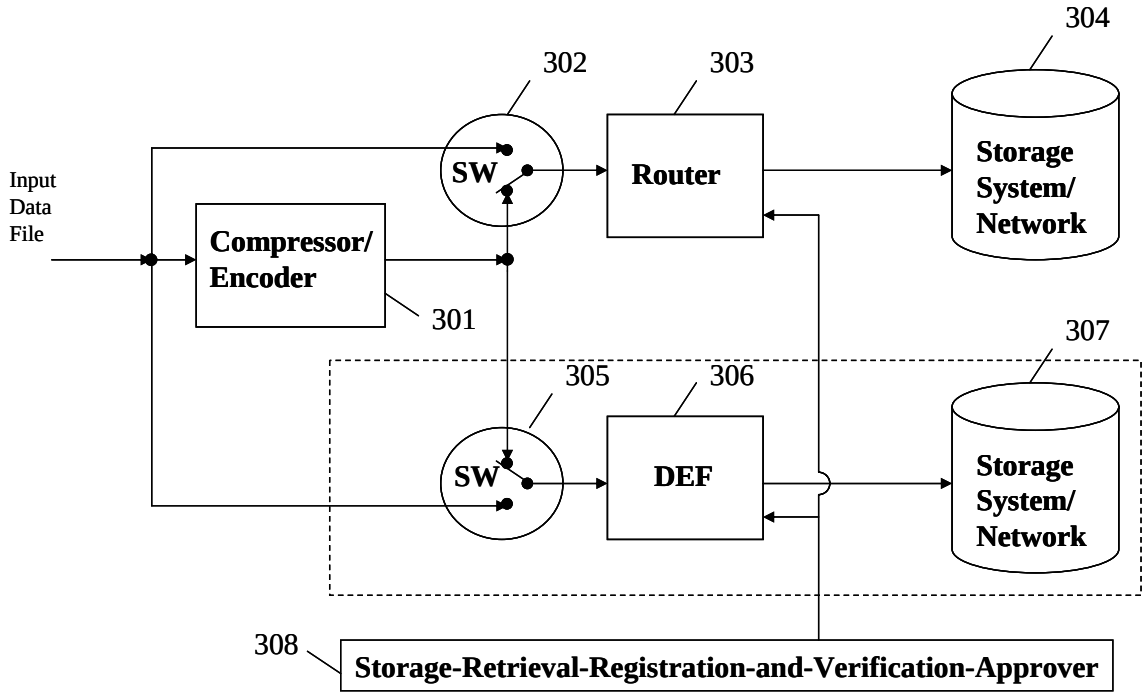

Figure 3 a). An Example application of the DEF apparatus; file storage.

This example serves to demonstrate how the DEF apparatus can be used for the purposes of data integrity measurement and similarity measurement. A scenario that can develop is one where Storage System/Network 304 is useful for the sharing of data and/or capable of large volumes of storage. However, due to the public nature of this network, the data is vulnerable to unauthorized changes, incorrect claims to ownership, unclear authenticity, or any combination of these and related problems. Through the policies and policing actions of Storage-Retrieval-Registration-and-Verification-Approver 308, the depicted process of fileprint utilization can detect changes and clarify ownership and authentication issues. Thus it is implied that Storage-Retrieval-Registration-and-Verification-Approver 308 is maintained at a security level that is equal to or higher than any given security level within the example.

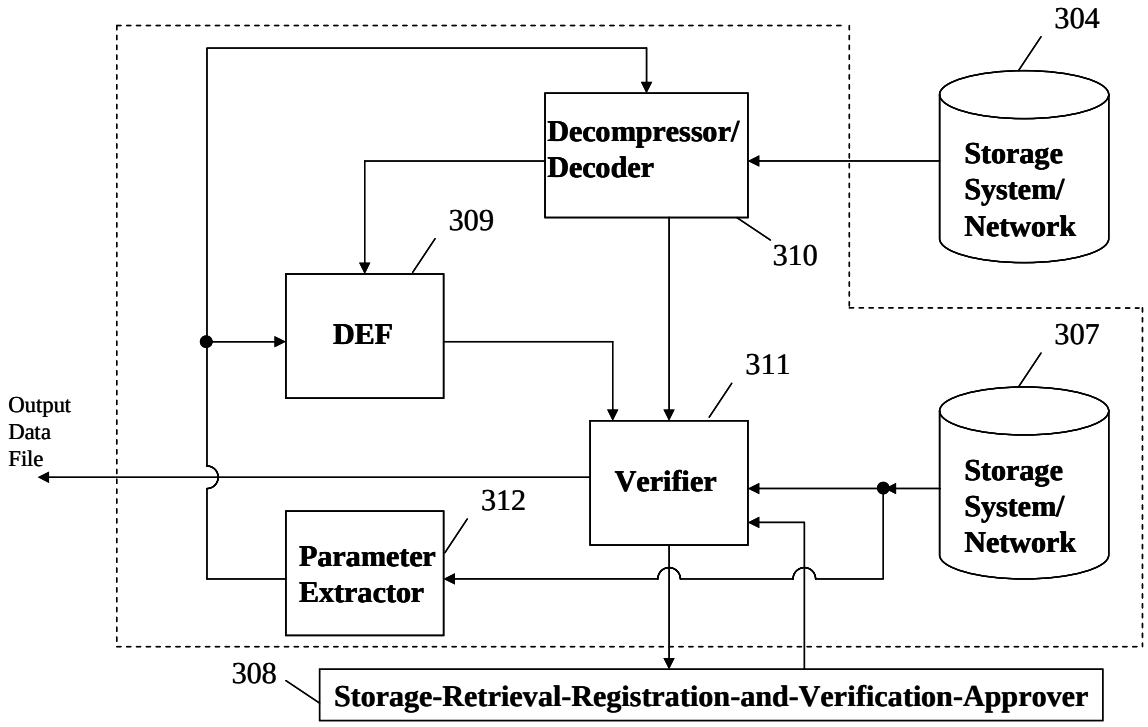

Figure 3 b). An example application of the DEF apparatus; file retrieval.

3.2 Fileprint Examples

The applicability of the DEF apparatus to the purpose of automatic identification of data type will be demonstrated by providing examples of fileprints from actual data files. The fileprints have been extracted using an experimental DEF implementation. These example fileprints demonstrate the fact that files of differing types, audio files with .wav extensions and image files with .bmp extensions, exhibit patterns in the fileprints which can be used to automatically identify the type of data file. The ability to measure similarity is also made apparent.

Fileprints were extracted from a pair of audio files, faks0_sa1.wav and faks0_sa1lsbmod.wav, and are shown in Figures 4(a) through 4(f). An ABC Processor with 3 stages was used in the DEF process. The first stage filter for averaging over frequency was a symmetric unity-gain FIR filter, designed using a Hanning window and 41 coefficients. The second stage filter for averaging over frequency was a symmetric unity-gain FIR filter, designed using a Hanning window and 11 coefficients. Circular convolution was employed, and no averaging over time was used in the stages. The first stage threshold was set to an estimate of the average noise power from the Periodogram Calculator, and is constant for the entire file. The second and third stage thresholds were each set to zero for the entire file. The parameter N was chosen to be 1024 and 8-bit samples were used. Segments consisted of N consecutive samples read from the file, with no overlap. All samples contained in each of the files were used.

To simulate unauthorized changes to the data file, the audio file faks0_sa1.wav was used to generate the same sized audio file, faks0_sa1lsbmod.wav, by setting all least-significant bits of the 16-bit audio samples to zero. The original audio file, faks0_sa1.wav is from the commonly used TIMIT database. Visual inspection of Figures 4(a) and 4(b) shows that the fileprints from stage 1 have strong similarity between the original and altered files. A more careful visual inspection of these figures also reveals slight differences appearing randomly within the fileprints. Such differences can be seen, for example, at the detection results near the locations (frequency bin 410, input segment 19) or (frequency bin 70, input segment 45). In a similar fashion, Figures 4(c) and 4(d) show similarities and differences between the stage 2 fileprints of both audio files. Likewise, Figures 4(e) and 4(f) show similarities and differences between the stage 3 fileprints of both audio files. As stage number increases, the differences between the compared fileprints tend to also increase. Therefore, a general trend is that stage 1 fileprints are more useful for presenting similarities, while stages 2 and 3 are more useful for presenting differences between the audio files.

Image file examples are also provided. Fileprints were extracted from a pair of image files, lena.bmp and lenalsbmod.bmp, and the first 62 input segments are shown in Figures 5(a) through 5(f). An ABC Processor with 3 stages was used in the DEF process. The first stage filter for averaging over frequency was a symmetric unity-gain FIR filter, designed using a Hanning window and 41 coefficients. The second stage filter for averaging over frequency was a symmetric unity-gain FIR filter, designed using a Hanning window and 11 coefficients. Circular convolution was employed, and no averaging over time was used in the stages. The first stage threshold was set to an estimate of the average noise power from the Periodogram Calculator, and is constant for the entire file. The second and third stage thresholds were each set to zero for the entire file. The parameter N was chosen to be 1024 and 8-bit samples were used. Segments consisted of N consecutive samples read from the file, with no overlap. All samples contained in each of the files were used to generate the fileprints, however, only the first 62 input segments are shown to allow visual comparisons to be made to the audio examples of Figures 4(a) through 4(f).

To simulate unauthorized changes to the data file, the image file lena.bmp was used to generate the same sized image file, lenalsbmod.bmp, by setting to zero all least-significant bits of the 8-bit RGB components of samples. The original image file, lena.bmp is the commonly used Lena image. Visual inspection of Figures 5(a) and 5(b) shows that the fileprints from stage 1 have strong similarity between the original and altered files. A more careful visual inspection of these figures also reveals slight differences appearing randomly within the fileprints. Such differences can be seen, for example, at the detection results near the locations (frequency bin 220, input segment 20) or (frequency bin 80, input segment 27). In a similar fashion, Figures 5(c) and 5(d) show similarities and differences between the stage 2 fileprints of both image files. Likewise, Figures 5(e) and 5(f) show similarities and differences between the stage 3 fileprints of both image files. As stage number increases, the differences between the compared fileprints tend to also increase. Therefore, a general trend is that stage 1 fileprints are more useful for presenting similarities, while stages 2 and 3 are more useful for presenting differences between the image files.

Note also that visual comparisons of the fileprints shown in the audio examples of Figures 4(a) through 4(f) to the fileprints shown in the image examples of Figures 5(a) through 5(f), allow for the observances of characteristics of the file types. In particular, note the increase in structure for the .bmp image fileprints near frequency bin 350. Observation of fileprints

extracted from additional .wav, .bmp and other file types have confirmed the consistency of characteristics unique to the file type.

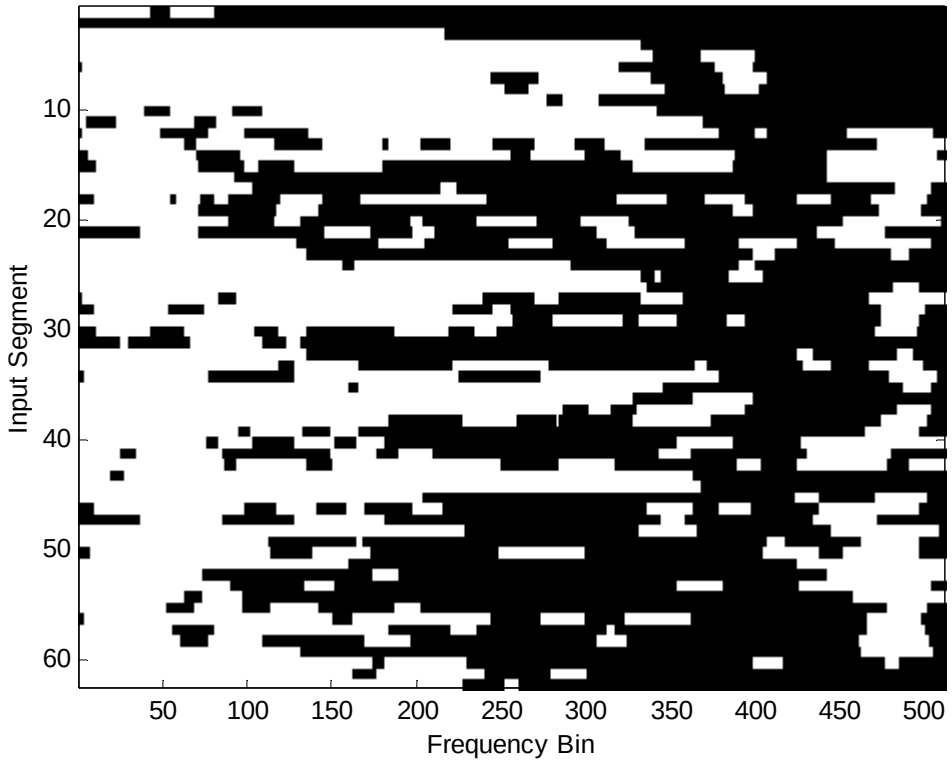

Figure 4(a). Stage 1 detections, a fileprint of file faks0_sa1.wav

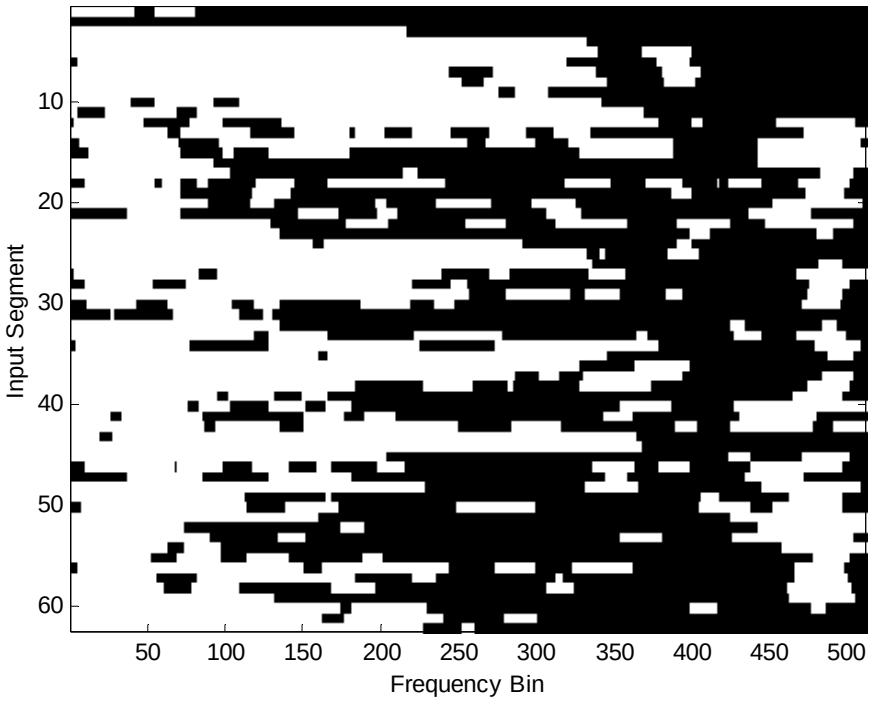

Figure 4(b). Stage 1 detections, a fileprint of file faks0_sa1lsbmod.wav

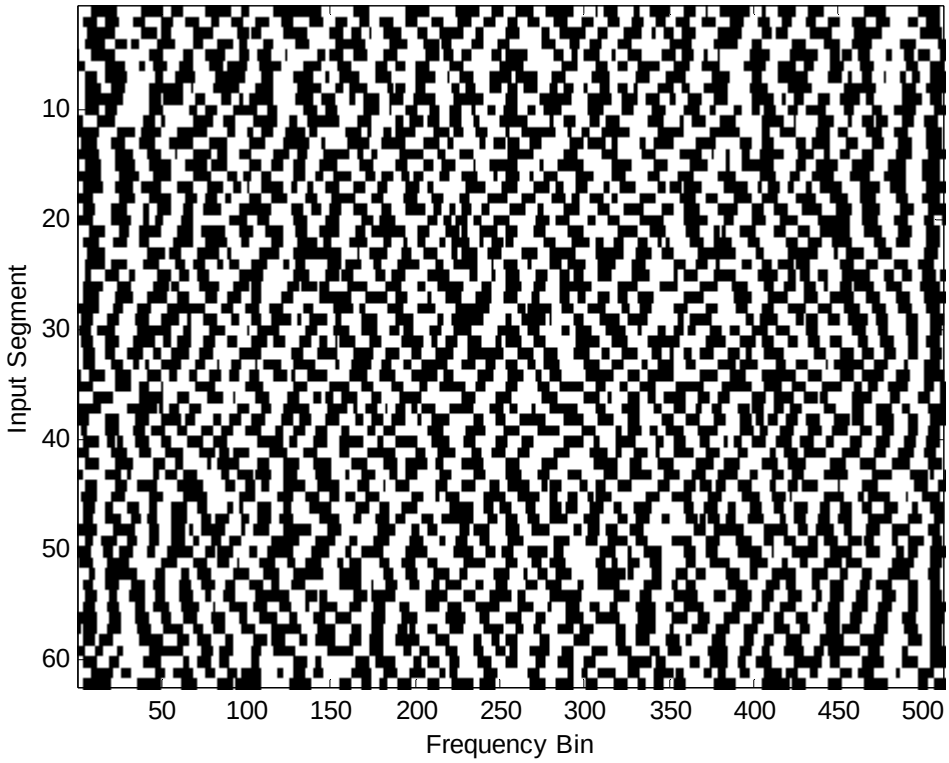

Figure 4(c). Stage 2 detections, a fileprint of file faks0_sa1.wav

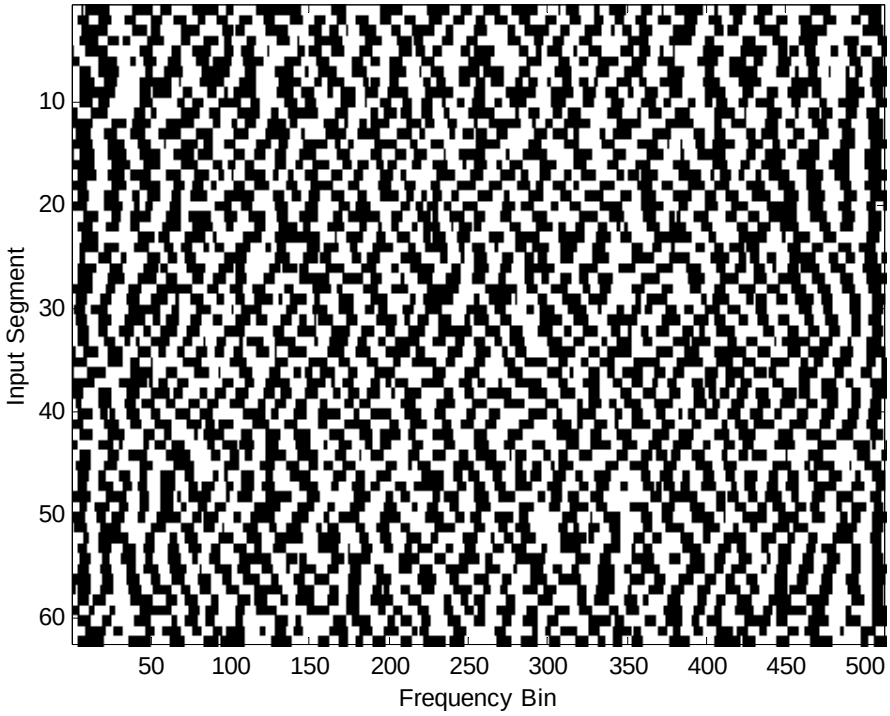

Figure 4(d). Stage 2 detections, a fileprint of file faks0_sa1lsbmod.wav

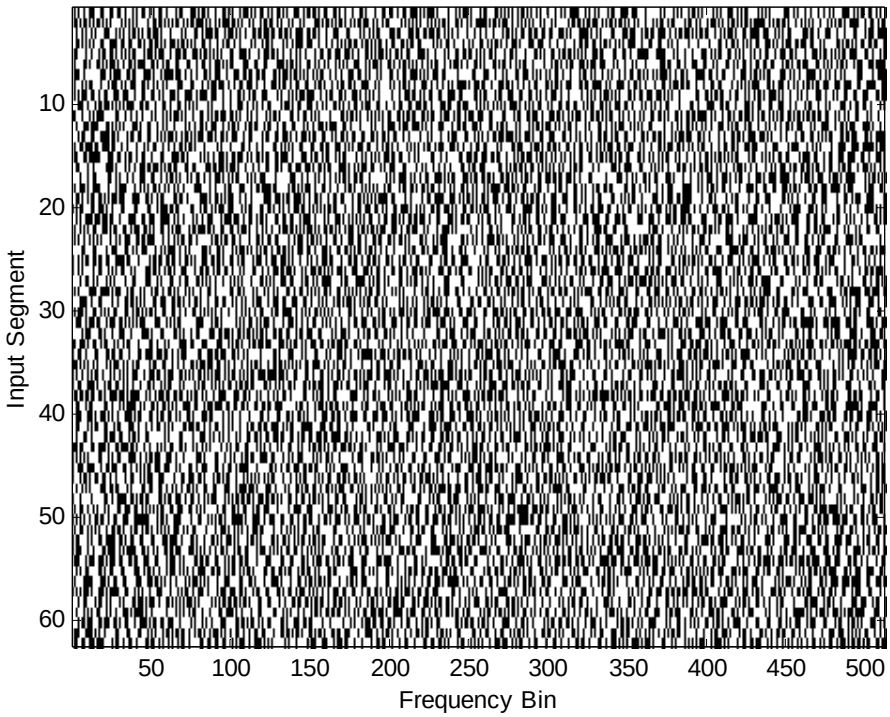

Figure 4(e). Stage 3 detections, a fileprint of file faks0_sa1.wav

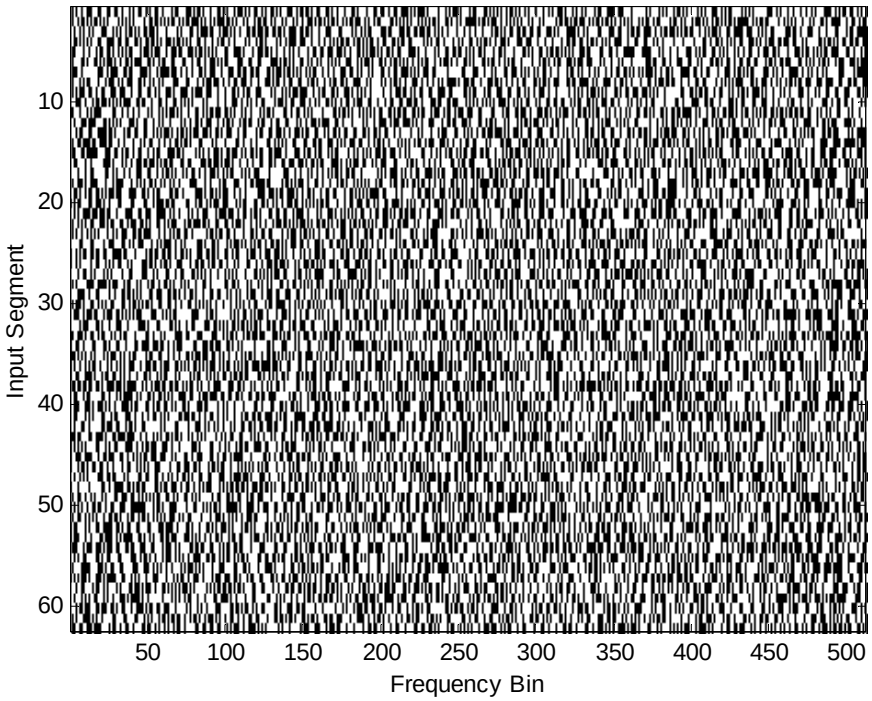

Figure 4(f). Stage 3 detections, a fileprint of file faks0_sa1lsbmod.wav

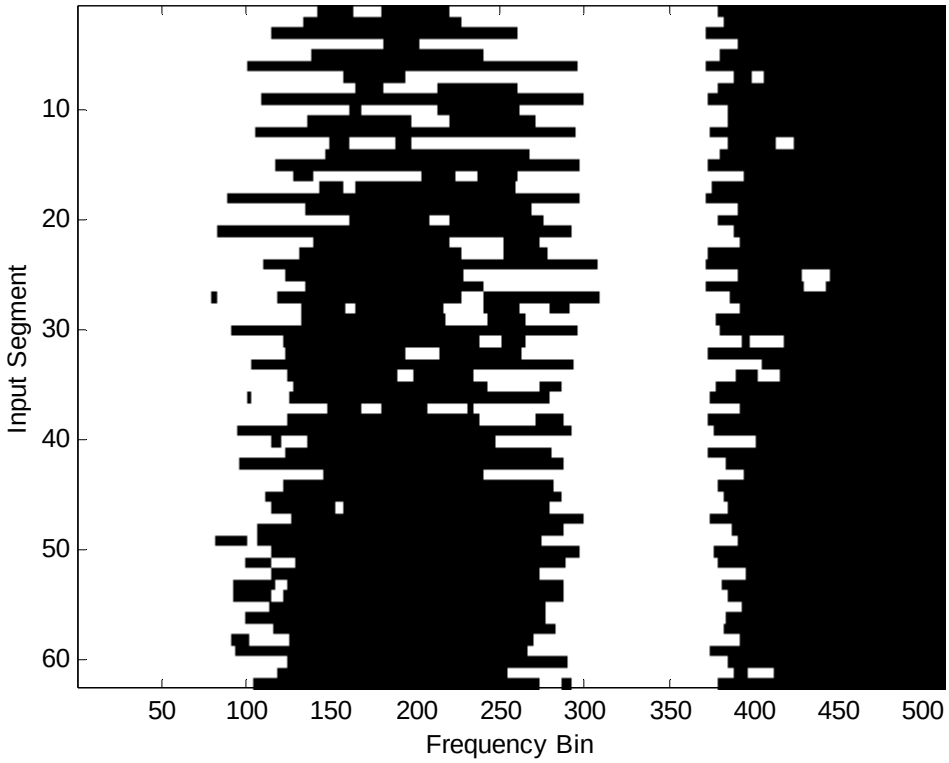

Figure 5(a). Stage 1 detections, a partial fileprint of file lena.bmp

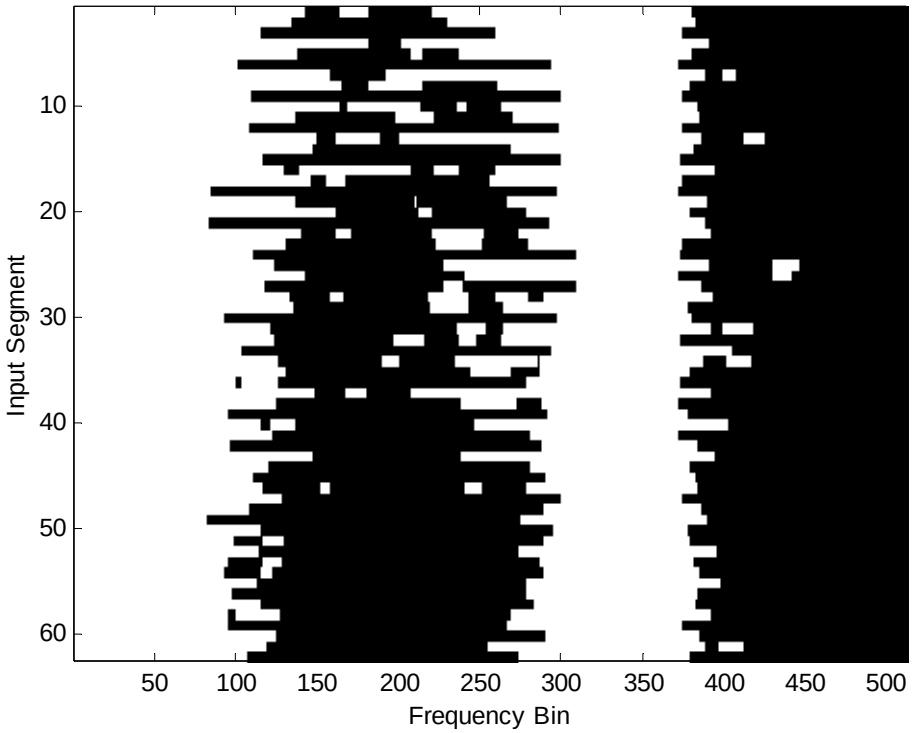

Figure 5(b). Stage 1 detections, a partial fileprint of file lenalsbmod.bmp

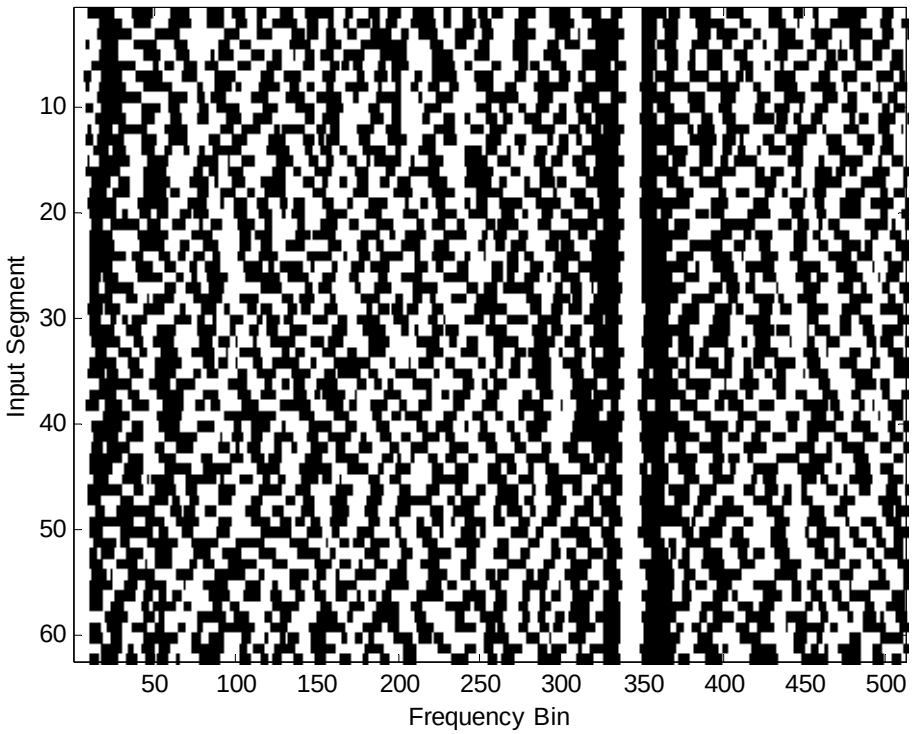

Figure 5(c). Stage 2 detections, a partial fileprint of file lena.bmp

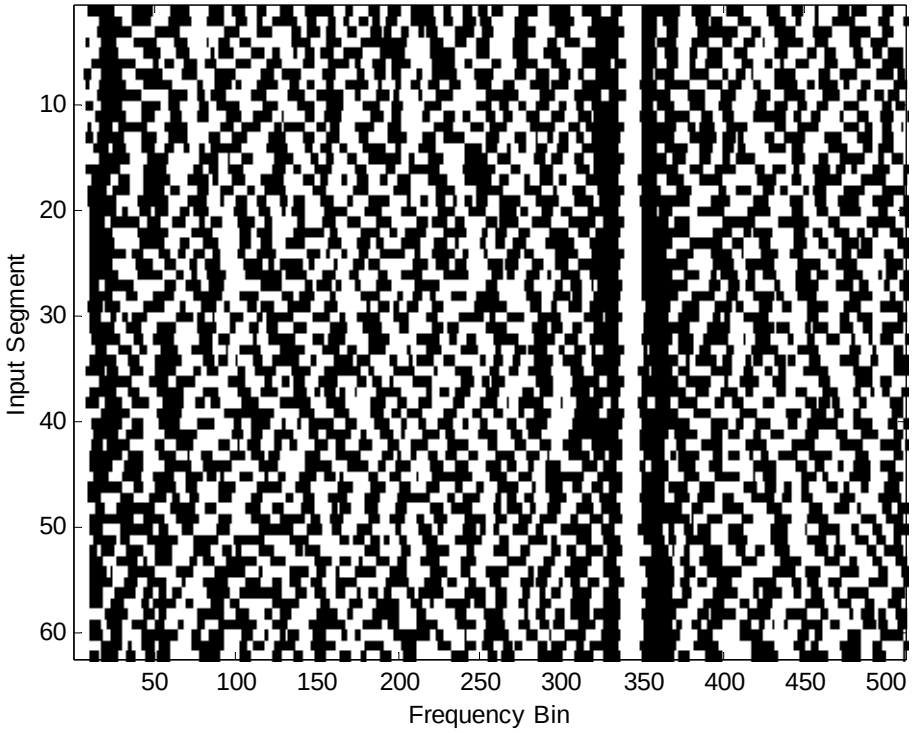

Figure 5(d). Stage 2 detections, a partial fileprint of file lenalsbmod.bmp

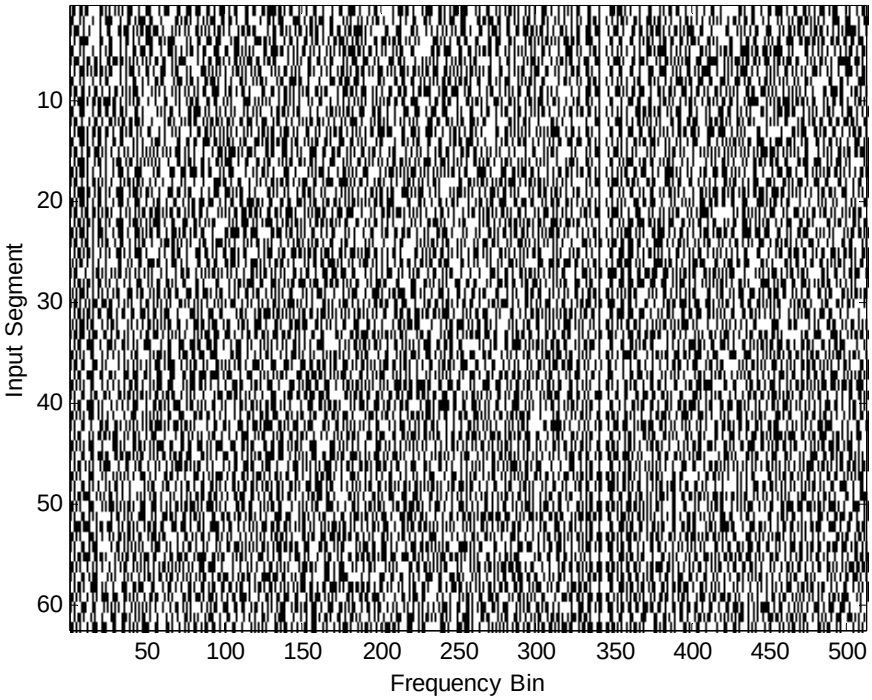

Figure 5(e). Stage 3 detections, a partial fileprint of file lena.bmp

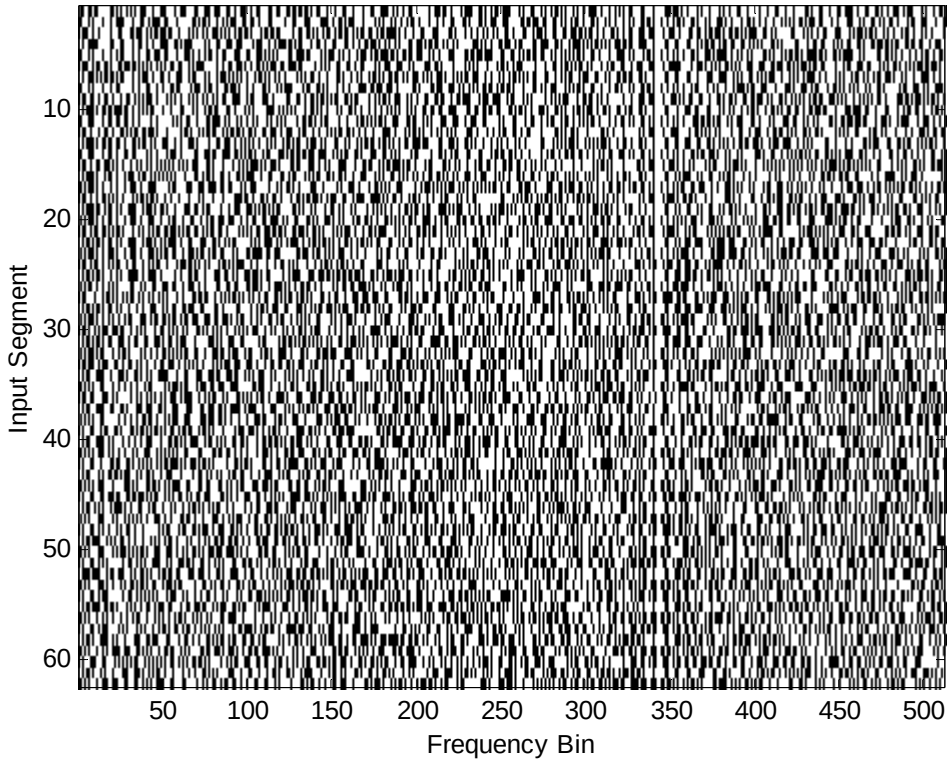

Figure 5(f). Stage 3 detections, a partial fileprint of file lenalsbmod.bmp

3.3 Advantages and New Features

There are several advantages attributable to the DEF apparatus relative to prior methods of data protection. An important advantage is the fact that no auxiliary data is embedded into the host data to be protected. This eliminates the processing time and computational resources normally required to embed the auxiliary data. Furthermore, the process of embedding auxiliary data will alter the original host data. The alteration of host data, particularly image and audio data, degrades the value of the host data. Because no auxiliary data is embedded into the host data by the DEF process, no degradation occurs.

A related advantage is due to the fact that the fileprints extracted from the original data are unique to the data. Therefore, the registration information need not include the original data. In cases where the data is considered proprietary and/or of high value, non-disclosure of the data can be an important issue.

Another advantage is the fact that unauthorized changes detected by the DEF process are associated with the segment in which the changes occurred. This allows for locating such changes and contributes to making decisions regarding the corrective course of action to be taken.

Yet another advantage is the fact that the transformation to the frequency domain within the DEF process produces measures of periodic components of the data. Format information associated with data transmission and storage can result in frequency domain representations that

are unique. Audio and image data also produce highly informative frequency domain representations. As a result, the fileprints extracted using the DEF process retain important characteristics that can be associated with data types. This allows for the characterization and identification of data sequences when a-priori knowledge is limited.

A further advantage attributable to the DEF process is the fact that parameters can be chosen to suit the requirements of the application. Important aspects such as input data parsing, frequency resolution and fileprint size can be controlled.

Still another advantage is achieved by using the DEF process in generating fileprints. The fileprints that are produced by the DEF process can be used to create binary images as presented in the examples of Figures 4(a) through 4(f) and 5(a) through 5(f). This allows for human visual inspection of the fileprints. The presentation and inspection of fileprints by humans can be useful in proceedings where evidence is required regarding data ownership and authenticity.

4.0 Conclusions

The flexibility in the choice of parameters for the DEF process leads to a large number of variations resulting in useful configurations. Enumeration of all useful configurations is not feasible. However, particularly important configurations are achieved by careful consideration of the parsing parameters.

The input data sequence in simplest form is a binary sequence. This input must be interpreted as a sequence of numbers, referred to as samples. Sample definition is included among the parsing parameters. A result is that fileprints can be extracted and maintained for original data sequences, compressed/encoded versions, or both. Examples using audio .wav files and image .bmp files have been presented (see also [9]). However, there is no limitation on the type of data that can be protected and can include for example, video data and executable code.

Parsing parameters are also used to form sequences of samples called segments. For applications requiring higher security, the starting locations of segments can be determined by a pseudo-random number generator. The seed for the pseudo-random number generator is retained as a parsing parameter in such configurations. Segments can be chosen with a random amount of overlap, normally such that the complete data sequence has been processed. Alternatively, segments can be formed with no overlap and/or with decimation such that processing speed is increased. This can also reduce fileprint storage requirements, but can reduce the effectiveness of the protection.

In addition to alternatives based on the choice of parameters, other useful modifications exist. In digital watermarking, the process of embedding auxiliary data can require resources and data quality can be degraded. However, the auxiliary data may be useful information. Also, the auxiliary information is embedded in a way known only to intended recipients. The DEF process can be used to enhance the transmission of auxiliary data. The auxiliary data can be appended to the host data without host data alterations, rather than embedding it into the host data with host data alterations. Fileprints of the resulting data sequence then be used to protect both the host and the appended auxiliary data. In this way, host data quality is not degraded. Although it may not be hidden, encryption can be used to secure the auxiliary data.

5.0 References

[1] A. J. Noga, "Adjustable Bandwidth Concept Signal Energy Detector," *U.S. Patent 5,257,211*, October 1993.

[2] A. J. Noga, "Adjustable Bandwidth Concept (ABC) Performance Evaluation," *Interim Report AFRL-IF-RS-TR-2003-184*, July 2003.

[3] B. Costello, "Graphical Interface Concept for a Signal Detection Process," *In-House Technical Memo AFRL-IF-RS-TM-2003-1*, February 2003.

[4] A. J. Noga, "Complex Band-pass Filters for Analytic Signal Generation and their Application," *In-House Technical Memo AFRL-IF-RS-TM-2001-1*, July 2001.

[5] C. I. Podilchuk, E. J. Delp, "Digital Watermarking: Algorithms and Applications" *IEEE Signal Processing Magazine*, Vol. 18, No. 4, July 2001 pp. 33-46.

[6] I. J. Cox, M. L. Miller, "The First 50 Years of Electronic Watermarking," *EURASIP Journal on Applied Signal Processing* 2002:2, pp. 126-132.

[7] R. Venkatesan, S.-M. Koon, M. H. Jakubowski, P. Moulin, "Robust Image Hashing," 2000 *IEEE International Conference on Image Processing*, ICIP00.

[8] J. G. Proakis, *Digital Communications*, 2nd Ed., McGraw-Hill Inc., 1989.

[9] J. Haitsma, T. Kalker, "A Highly Robust Audio Fingerprinting System," *IRCAM 2002*.